Outline

• The Nisan-Wigderson generator
• Error correcting codes from polynomials
• Turning worst-case hardness into average-case hardness

Hardness vs. randomness

• We have shown:
  If one-way permutations exist then
  \( \text{BPP} \subseteq \cap_{f:0,1^l\rightarrow 0,1^l} \text{TIME}(2^{n^\delta}) \subseteq \text{EXP} \)
• simulation is better than brute force, but just barely
• stronger assumptions on difficulty of inverting OWF lead to better simulations…

Hardness vs. randomness

• We will show:
  If \( \textbf{E} \) requires exponential size circuits then
  \( \text{BPP} = \text{P} \)
  by building a different generator from different assumptions.
  \[ \text{E} = \cup_k \text{DTIME}(2^{kn}) \]

Hardness vs. randomness

• BMY: for every \(\delta > 0\), \(G^3\) is a PRG with
  seed length \(t = m^\delta\)
  output length \(m\)
  error \(\epsilon < 1/m^d\) (all \(d\))
  fooling size \(s = m^e\) (all \(e\))
  running time \(m^c\)
• running time of simulation dominated by \(2^t\)
**Hardness vs. randomness**

- **BMY pseudo-random generator:**
  - one generator fooling all poly-size bounds
  - one-way-permutation is hard function
  - implies hard function in $\text{NP} \cap \text{coNP}$
- **New idea (Nisan-Wigderson):**
  - for each poly-size bound, one generator
  - hard function allowed to be in $E = \cup \text{DTIME}(2^{\text{poly}(n)})$

**Comparison**

<table>
<thead>
<tr>
<th>BMY: $\forall \delta &gt; 0$ PRG $G^\delta$</th>
<th>NW: PRG $G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>seed length $t = m^\delta$</td>
<td>$t = O(\log m)$</td>
</tr>
<tr>
<td>running time $t^*m$</td>
<td>$m^\delta$</td>
</tr>
<tr>
<td>output length $m$</td>
<td>$m$</td>
</tr>
<tr>
<td>error $\epsilon &lt; 1/m^\delta$ (all d)</td>
<td>$\epsilon &lt; 1/m$</td>
</tr>
<tr>
<td>fooling size $S = m^\delta$ (all e)</td>
<td>$S = m$</td>
</tr>
</tbody>
</table>

**NW PRG**

- NW: for fixed constant $\delta$, $G = \{G_n\}$ with
  - seed length $t = O(\log n)$
  - running time $m^\delta$  
  - output length $m$
  - error $\epsilon < 1/m$
  - fooling size $S = m$
  - Using this PRG we obtain $\text{BPP} = \text{P}$
    - to fool size $n^\delta$ use $G_{\log n}$
    - running time $O(n^\delta + n^{\delta/2})^2 = \text{poly}(n)$

**NW PRG**

- First attempt: build PRG assuming $E$ contains unapproximable functions

**Definition:** The function family

$f = \{f_n\}, f_n: \{0,1\}^n \to \{0,1\}$

is $s(n)$-unapproximable if for every family of size $s(n)$ circuits $\{C_n\}$:

$\Pr_x[C_n(x) = f_n(x)] \leq 1/2 + 1/s(n)$.

**One bit**

- Suppose $f = \{f_n\}$ is $s(n)$-unapproximable, for $s(n) = 2^{O(n)}$, and in $E$
- a “1-bit” generator family $G = \{G_n\}$:
  
  $G_n(y) = y^*_f \log n(y)$

- Idea: if not a PRG then exists a predictor that computes $f_{\log n}$, with better than $1/2 + 1/s(\log n)$ agreement; contradiction.

**One bit**

- Suppose $f = \{f_n\}$ is $s(n)$-unapproximable, for $s(n) = 2^n$, and in $E$
- a “1-bit” generator family $G = \{G_n\}$:

  $G_n(y) = y^*_f \log n(y)$

  - seed length $t = \log n$
  - output length $m = \log n + 1$ (want $n^\delta$)
  - fooling size $S = s(\log n) = n^\delta$
  - running time $n^\delta$
  - error $\epsilon = 1/s(\log n) = 1/n^\delta < 1/m$
Many bits

- Try outputting many evaluations of $f$:
  \[ G(y) = f(b_1(y)) \cdot f(b_2(y)) \cdot \ldots \cdot f(b_m(y)) \]

- Seems that a predictor must evaluate $f(b_i(y))$ to predict $i$-th bit

- Does this work?

Nearly-Disjoint Subsets

**Definition:** $S_1, S_2, \ldots, S_m \subset \{1 \ldots t\}$ is an $(h, a)$ design if
- for all $i$, $|S_i| = h$
- for all $i \neq j$, $|S_i \cap S_j| \leq a$

**Lemma:** for every $\epsilon > 0$ and $m < n$ can in poly(n) time construct an
$(h = \log n, a = \epsilon \log n)$ design $S_1, S_2, \ldots, S_m \subset \{1 \ldots t\}$ with $t = O(\log n)$.

The NW generator

- $f \in \mathbb{E}$ $s(n)$-unapproximable, for $s(n) = 2^{\Theta(n)}$
- $S_1, \ldots, S_m \subset \{1 \ldots t\}$ $(\log n, a = \delta \log n/3)$ design with $t = O(\log n)$
  \[ G_n(y) = f_{\log n}(y_{S_1}) \cdot f_{\log n}(y_{S_2}) \cdot \ldots \cdot f_{\log n}(y_{S_m}) \]
- $f_{\log n}$ is a fixed function with $f_{\log n}(y) = 01010010111010101110010010$
The NW generator

**Theorem** (Nisan-Wigderson): $G = \{G_n\}$ is a pseudo-random generator with:
- seed length $t = O(\log n)$
- output length $m = n^{\beta/3}$
- running time $n^c$
- fooling size $S = m$
- error $\epsilon = 1/m$

**Proof (continued):**
- fix bits outside of $S_i$ to preserve advantage:
  $Pr_y[P(G_n(y), \ldots, y_{i-1}) = G_n(y)_i] > \frac{1}{2} + \epsilon/m$

Worst-case vs. Average-case

**Theorem** (NW): if $E$ contains $2^{O(n)}$-unapproximable functions then $BPP = P$.

• How reasonable is unapproximability assumption?
• Hope: obtain $BPP = P$ from worst-case complexity assumption
  - try to fit into existing framework without new notion of “unapproximability”
Worst-case vs. Average-case

**Theorem** (Impagliazzo-Wigderson, Sudan-Trevisan-Vadhan)
If \( E \) contains functions that require size \( 2^{\Omega(n)} \) circuits, then \( E \) contains \( 2^{\Omega(n)} \)-unapproximable functions.

- **Proof:**
  - main tool: error correcting code

Distance and error correction

- \( C \) is an ECC with distance \( d \)
- can uniquely decode from up to \( \lfloor d/2 \rfloor \) errors

Example: Reed-Solomon

- alphabet \( \Sigma = F_q \): field with \( q \) elements
- message \( m \in \Sigma^k \)
- polynomial of degree at most \( k-1 \)
  \[ p_m(x) = \sum_{i=0}^{k-1} m_i x^i \]
- codeword \( C(m) = (p_m(x))_x \in F_q \)
- rate \( k/q \)

Error-correcting codes

- **Error Correcting Code (ECC):**
  \[ C : \Sigma^n \rightarrow \Sigma^k \]
  - message \( m \in \Sigma^k \)
  - received word \( R \)
    - \( C(m) \) with some positions corrupted
  - if not too many errors, can decode: \( D(R) = m \)
  - parameters of interest:
    - rate: \( k/n \)
    - distance:
      \[ d = \min_{m,m'} \Delta(C(m), C(m')) \]

Distance and error correction

- can find short list of messages (one correct) after closer to \( d \) errors!

**Theorem** (Johnson): a binary code with distance \( (\frac{1}{2} - \delta)n \) has at most \( O(1/\delta^3) \) codewords in any ball of radius \( (\frac{1}{2} - \delta)n \).

Example: Reed-Solomon

- **Claim:** distance \( d = q - k + 1 \)
  - suppose \( \Delta(C(m), C(m')) < q - k + 1 \)
  - then there exist polynomials \( p_m(x) \) and \( p_{m'}(x) \) that agree on more than \( k-1 \) points in \( F_q \)
  - polynomials \( p(x) = p_m(x) \cdot p_{m'}(x) \) has more than \( k-1 \) zeros
  - but degree at most \( k-1 \)...
  - contradiction.
Example: Reed-Muller

- **Parameters:** \( t \) (dimension), \( h \) (degree)
- alphabet \( \Sigma = F_q \): field with \( q \) elements
- message \( m \in \Sigma^t \)
- multivariate polynomial of total degree at most \( h \):
  \[
p_m(x) = \sum_{i=0}^{h-1} m_i M_i
\]
  \( \{M_i\} \) are all monomials of degree \( \leq h \)

---

Codes and hardness

- Reed-Solomon (RS) and Reed-Muller (RM) codes are efficiently encodable
- **efficient unique decoding?**
  - yes (classic result)
- **efficient list-decoding?**
  - yes (recent result: Sudan. On problem set.)

---

Codes and Hardness

- Use for worst-case to average case:
  - truth table of \( f : \{0,1\}^{\log k} \rightarrow \{0,1\} \)
    - (worst-case hard)
    - \( m : 01100010 \)
    - truth table of \( f' : \{0,1\}^{\log n} \rightarrow \{0,1\} \)
      - (average-case hard)
      - \( C(m) : 01100010000010 \)

---

Codes and Hardness

- if \( n = \text{poly}(k) \) then
  \( f \in E \) implies \( f' \in E \)
- Want to be able to prove:
  - if \( f' \) is \( s'\)-approximable,
    then \( f \) is computable by a size \( s = \text{poly}(s') \) circuit

---

Codes and Hardness

- Key: circuit \( C \) that approximates \( f \) implicitly gives received word \( R \)
  \[
  R : 01101000100010
  \]
  \[
  C(m) : 01100010000010
  \]
- Decoding procedure \( D \) “computes” \( f \) exactly
  - Requires special notion of efficient decoding